SDP ASSIGNMENT TASK 3

Name : Rahat Hossain
Roll : 32

Problem Description:
The problem is to design a modern User Interface (UI) that supports multiple design styles (motif or ”look and feel”). The core of the UI is the single Window Manager (WM), which is responsible for the management of the UI items, such as buttons, text boxes and edit boxes. Different design styles are supported by the system: simplistic design style(for example, we can change the color of the elements), high detailed design style(for example, we can change the color and text size of the elements). WM should be initialized only by specifying the design style. Each UI item looks differently when the design style of the system is different. Also, each item has a value, which is displayed on the item itself (i.e. value of a button is the text displayed on the button).
The structure of a UI is described in a special config file. This file contains the structure of a UI as a list of UI items, their values and their coordinates. Example of a config file:
Button, Click on me, X: 250, Y: 300
EditBox, Some text to edit, X: 250, Y: 350
…
A Config Manager class is responsible for loading config files. It has methods nextItem() - returns the next item in the list and hasMoreItems() - returns true if iterating through the list is not over yet. Window Manager has a method loadUI(ConfigManager config), which goes through the config step by step and displays all UI items.
To extend the functionality of the system, it must also be possible to load the configuration from an XML file. Your application should adapt one of the native java XML parsing methods (DOM Parser/Builder, SAX Parser, Java XML I/O) and make it compatible with the ConfigManager interface. Example of an XML file:
<Button value=“Click on me” X=“250” Y=“300” />
<EditBox value=“Some text to edit” X=”250” Y=“350” />
...

Finally, to test the system, load UI elements from a config file, then from an XML file and then create several items during the runtime programmatically.

Project stack:

- Javascript (nodejs)

Classes/Components used:

- WindowManager: Singleton class that produces necessary components for visualization
- ConfigManager: Abstract super class for different types of config managers.
- ConfigManagerFile: ConfigManager class for file system.
- ConfigManagerXML: ConfigManagerClass for XML system
- AbstractFactory: AbstractFactory class for wrapping configManager Class
- Util: Utility class.
- Client/main: driver code.
- WindowManagerAdapter: Acts as an adapter between windowManager and ConfigManager.

Assumptions:

- WindowManager
    - Acts like a singleton class.
    - Stores its instance for future references.
    - Gets ‘designType’ as argument.
    - Specifies what type of component property to use based on designType.
    - Takes config manager, converts all elements to corresponding html syntax with styling and stores them for future use.
- ConfigManager:
    - Acts like a superclass.
    - read() method is kept for subclass implementation. Makes it compatible with different objectives/methods.
    - hasMoreItem(), nextItem() methods along with other necessary variables are declared and specified for subclasses.
- ConfigManagerFile:
    - Implements read() method itself.
- ConfigManagerXML:
    - Implements read() method itself.
- AbstractFactory:
    - Takes fileType and filename as argument.
    - Serves config manager as a factory item.
- WindowManager:
    - Acts as an adapter between configManager and windowManager.
- Util:
    - Implements utility functions for taking input and showing output.
- Client/Main:
    - Acts like the driver code.
    - Takes all necessary input from the user on the console.
    - Initiates server and serves the visual generated by the system.

Overall:

The Adapter, the Singleton and the Abstract Factory pattern have been used to build the solution.